Eötvös Loránd Tudományegyetem
Természettudományi Kar
Operációkutatási tanszék

Kriptográfiai protokollok

Villányi Viktória Ildikó
Egyetemi adjunktus

Szabó Zsuzsanna
Elemző matematikus BSc

Budapest, 2017

Tartalomjegyzék

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Villányi Viktóriának az egész féléves munkáját, hogy ötleteivel és szakértelmével hozzájárult a szakdolgozatom elkészítéséhez.
Továbbá szeretném megköszönni a családomnak és barátaimnak, hogy az egyetem évei alatt végig segítettek és támogattak.

Bevezetés

A kriptográfia szerepe sokat változott az évszázadok során. A kezdetekben inkább egy művészeti ágként tekintettek rá és leginkább a két fél között folyó kommunikáció titkosítása volt a cél. A jól használható kódok megalkotása és már meglévők feltörése kreativitást és egyéni készségeket igényelt. A 20. században a kriptográfiáról alkotott vélemény megváltozott, túlnőtte addigi jelentését. A technológia fejlődése lehetővé tette például az üzenetek hitelesítését, digitális aláírások létrehozását, titkos kulcsok megosztását, digitális pénzzel való kereskedést és elektronikus választások és aukciók létrehozását. Tökéletes definíciót nem lehet alkotni a kriptográfiára, de manapság úgy tekintünk rá, mint a digitális információ, tranzakció, számítások biztonságával foglalkozó tudományág. Az évek során a felhasználók csoportja is megváltozott, kibővült. Eredetileg a katonaságnál volt nagy szerepe a kriptográfiának, manapság minden számítógépes rendszernél használjuk.

A kriptográfia mellett fontos tudományág a kriptoanalízis is, ami a kriptorendszerek feltörésével foglalkozik. Fontos, hogy ez a rész is szerepet kapjon a tudományok között, hiszen ha senki sem foglalkozna a már meglévő protokollok feltörésével, nem tudnánk megmondani, hogy biztonságos-e az adott eljárás.

A kriptográfiát két részre tudjuk bontani [4] :
Szimmetrikus algoritmusok: Egy titkosításból és egy dekódolásból állnak, ami alatt a két fél megoszt egymással egy titkos kulcsot. Ezeket az algoritmusokat a mai napig alkalmazzák, főleg adattitkosításnál és üzenetek eredetiségvizsgálatánál.
Aszimmetrikus algoritmusok: a felhasználók a titkos kulcson kívül egy publikus kulcsot is felhasználnak a titkosítás során. Leggyakrabban digitális aláírásoknál, kulcscserénél, illetve klasszikus adattitkosításnál használjuk ezt a módszert.

A gyakorlatban a szimmetrikus és aszimmetrikus sémákat sokszor együtt használják, ezeket hívjuk hibrid sémáknak. Erre azért van szükség, mert mindkét sémának megvan a maga gyengesége és erőssége, a két algoritmusfajta kombinálásával pedig bizonyos gyengeségeket ki lehet küszöbölni.

1. Matematikai alapok

1. Definíció (Kvadratikus maradék). *[1] Legyen $p > 2$ prím és $(a, p) = 1$. Az a számot aszerint nevezzük kvadratikus maradéknak, illetve kvadratikus nemmaradéknak modulo p, hogy az $x^2 \equiv a \mod p$ kongruencia megoldható-e, vagy sem.*

2. Definíció (Legendre-szimbólum). *[1] Az $(\frac{a}{p})$ Legendre-szimbólumot a következőképpen értelmezzük:*
$(\frac{a}{p}) = 1$, ha a kvadratikus maradék $\mod p$
$(\frac{a}{p}) = -1$ ha a kvadratikus nemmaradék $\mod p$.

3. Definíció (Jacobi-szimbólum). *[1] Legyen $m > 1$ páratlan szám, $m = p_1 ... p_r$, ahol a p_i számok (nem feltétlenül különböző) pozitív prímek. Legyen továbbá $(a, m) = 1$. Ekkor az $(\frac{a}{m})$ Jacobi-szimbólumot mint az $(\frac{a}{p_i})$ Legendre-szimbólumok szorzatát értelmezzük:*

$$(\tfrac{a}{m}) = (\tfrac{a}{p_1}) ... (\tfrac{a}{p_r}).$$

1. Tétel (Kínai maradéktétel). *[1] Legyenek az $m_1, m_2, ..., m_k$ modulusok páronként relatív prímek. Ekkor az*

$$\begin{aligned} x &\equiv c_1 \pmod{m_1} \\ x &\equiv c_2 \pmod{m_2} \\ &\vdots \\ x &\equiv c_k \pmod{m_k} \end{aligned}$$

szimultán kongruenciarendszer bármilyen c_1, c_2,...,c_k egészek esetén megoldható, és a megoldások egyetlen maradékosztályt alkotnak modulo $m_1 m_2 \cdots m_k$.

4. Definíció (Csoport). *[2] Egy $\mathbb{G}$ nemüres halmazt csoportnak nevezünk, ha értelmezve van $\mathbb{G}$-n egy asszociatív művelet, $\exists$ egységelem és $\forall$ elemnek van inverze.*

5. Definíció (Ciklikus csoport). *[2] Egy csoport ciklikus, ha egyetlen elem (összes egész kitevőjű) hatványaiból áll. Egy ilyen elemet a ciklikus csoport generátorelemének nevezzük.*

Például: egész számok összeadásra, modulo m maradékosztályok összeadásra, ha p prím, akkor modulo p redukált maradékosztályok multiplikatív csoportja. A g által generált ciklikus csoportot $\langle g \rangle$-vel jelöljük. $|\langle g \rangle| = o(g)$, így ha $o(g) = n$, akkor $\langle g \rangle = \{e, g^1, g^2, ..., g^{n-1}\}$. Ha $o(g) = \infty$, akkor a g-nek $\forall$ hatványa különböző.

6. Definíció (Galois mező). *[5] A Galois mező egy véges elemekből álló halmaz, amiben végrehajtható az összeadás, kivonás, szorzás és invertálás.*

2. Tétel (A "kis" Fermat-tétel). *[1] Ha p prím és $(a, p) = 1$, akkor $a^{p-1} \equiv q \pmod{p}$.*

2. Kriptográfiai bevezetés

2.1. Prímszámok keresése [1][2]

Látszólag könnyű meghatározni egy szám prímtényezős felbontását, hiszen csak végig kell nézni, hogy osztható-e 2-vel, 3-al, 5-el és így tovább. Ha találtunk egy osztót, akkor a hányadoson kell folytatni az eljárást. Ha az eredeti számunk négyzetgyökéig elmentünk és nem találtunk osztót, akkor a szám biztosan prím. Ez az algoritmus és az ehhez hasonló próbaosztásos algoritmusok egyszerűek, azonban nem hatékonyak, mivel nagyon nagy a számítási igényük. Azonban léteznek olyan algoritmusok, amik majdnem teljes pontossággal és relatív gyorsan megmondják egy számról, hogy prímszámról van-e szó. Ezeket az algoritmusokat nevezzük prímteszteknek.
A különbség az, hogy nem prímosztót keresnek, hanem olyan feltételeket vizsgálnak meg, amik a prímszámokra igazak, az összetett számokra viszont nagy valószínűséggel nem. Arra figyelni kell, hogy bár kis számban, de előfordulhatnak kivételek, ezt minden esetben figyelembe kell vennünk.

2.1.1. Speciális esetek: a Fermat- és Mersenne-prímek

Fermat-prímek

Fermat azt hitte, hogy az $F_n = 2^{2^n} + 1$ Fermat számok mind prímek, Euler azonban megmutatta, hogy $F_5 = 2^{32} + 1$ összetett szám. $n > 4$-re egyelőre nem találtak prímet az F_n számok között, annyit viszont biztosan tudunk, hogy F_n , $5 \leqslant n \leqslant 23$ esetén összetett szám. A Fermat-számok vizsgálatánál két tételt használhatunk, az egyik a prímosztó keresést könnyíti meg, a másik tétel pedig segít ellenőrizni, hogy az adott Fermat-szám prím-e.

3. Tétel. *F_n bármely (pozitív) osztója $k2^{n+1} + 1$, sőt $n \geqslant 2$ esetén $r2^{n+2} + 1$ alakú.*

4. Tétel (Pepin-teszt). *Az $n \geqslant 1$ esetben F_n akkor és csak akkor prím, ha $3^{(F_n-1)/2} \equiv -1 \mod F_n$.*

Mersenne-prímek

Legyen p prím. Ekkor az $M_p = 2^{p-1}$ számokat nevezzük Mersenne-számoknak. A Mersenne-számok sem lesznek mindig prímek, a legkisebb összetett szám p =11 esetén kapjuk. A Fermat-számokra vonatkozó tételek megfelelői Mersenne-számokra az alábbiak:

5. Tétel. *Legyen $p > 2$ prím. Ekkor M_p bármely (pozitív) osztója egyszerre $2kp + 1$ és $8r \pm 1$ alak.*

6. Tétel (Lucas-Lehmer-teszt). *Legyen $p > 2$ prím, továbbá $a_1 = 4$ és $a_{i+1} = a_i^2 - 2$, ha $i \geqslant 1$. Ekkor M_p pontosan akkor prím, ha: $M_p \mid a_{p-1}$.*

2.1.2. Prímtesztek

A legegyszerűbb általános prímteszt a kis Fermat-tételből következik:

7. Tétel. *Ha egy $n > 2$ számra $2^{n-1} \not\equiv 1 \mod n$, akkor n összetett. Ha $2^{n-1} \equiv 1 \mod n$, akkor n majdnem biztosan prím. A feltétel gyorsan ellenőrizhető, ha a hatványozást ismételt négyzetre emelések segítségével végezzük.*

Ez a teszt azonban nem alkalmas az álprímek kiszűrésére.

7. Definíció. *Ha egy n összetett számra $a^{n-1} \equiv 1 \mod n$ teljesül, akkor az n-et a a alapú álprímnek nevezzek.*
Ha az n összetett számra a fenti kongruencia minden $(a, n) = 1$ esetén teljesül, akkor az n univerzális álprím vagy Carmichael-szám.

A másik két prímteszt amit ebben a fejezetben megemlítek, már le tudják leplezni az álprímeket. Mindkettőben véletlen számokat használnak, amit egy generátor segítségével tudnak megválasztani.

8. Tétel (Solovay-Strassen-prímteszt). *Legyen $n > 1$ páratlan szám, és tekintsük az*

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \mod n$$

kongruenciát, ahol $\left(\frac{a}{n}\right)$ a Jacobi-szimbólum.
Ha n prím, akkor a kongruencia minden $a \not\equiv 0 \mod n$ esetén teljesül.
Ha n összetett, akkor a kongruencia egy modulo n teljes maradékrendszer elemeinek kevesebb, mint felére teljesül.
Megjegyzés: Ha $(a, n) > 1$, akkor az $\left(\frac{a}{n}\right)$ Jacobi-szimbólum nem értelmes, tehát a kongruencia nem teljesülhet.

9. Tétel (Miller-Lenstra-Rabin-prímteszt). *Legyen $n > 1$ páratlan szám, $n - 1 = 2^k r$, ahol r páratlan. Az*

$$a^r, a^{2r}, a^{4r}, \ldots, a^{2^{k-2}r} = a^{\frac{n-1}{4}}, a^{2^{k-1}r} = a^{\frac{n-1}{2}}$$

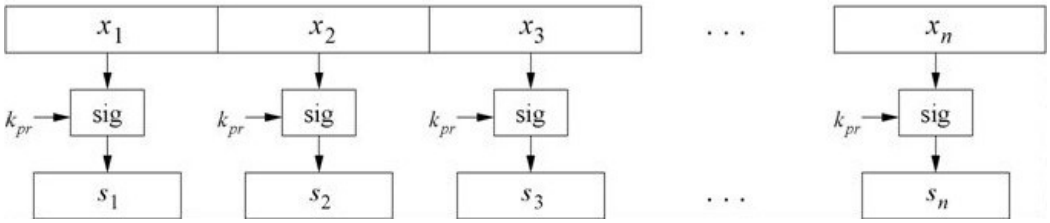
számokat jó sorozatnak nevezzük, ha ezek modulo n vett legkisebb abszolút értékű maradékai között előfordul -1 vagy pedig a^r maradéka 1.
Ha n prím, akkor sorozatunk minden $a \not\equiv 0 \mod n$ esetén jó sorozat.
Ha n összetett, akkor a sorozat egy modulo n teljes maradékrendszer elemeinek kevesebb, mint a felére alkot jó sorozatot.

2.2. HASH függvények [5]

A hash függvények fontos és gyakran használt függvényei a protokolloknak. Egy üzenetet tömörítenek, ami így egy rövidebb, adott hosszú bitsztring lesz. Úgy is tekinthetünk rájuk, mint az üzenet egy lenyomatára. Lényeges részt képeznek különböző aláíró sémákban, üzenet hitelesítéseknél, jelszó tárolásnál és kulcs képzésnél.

2.2.1. Hosszú üzenetek aláírása

Üzenetek aláírásánál figyelembe kell venni, hogy az elküldendő x üzenet terjedelmes és ilyenkor az aláírás számítási igénye nagyon nagy. Az első ötlet ami eszünkbe juthat a hatékonyság növelésére, hogy blokkokra bontjuk fel az aláírandó szöveget és az egyes blokkokat írjuk alá. Ez a módszer azonban több szempontból sem megfelelő:



[5]

Nagy számítási igény: aláírásnál a legegyszerűbb műveleteknek is van némi számítási idejük. Nagyobb üzenetek aláírásánál ez az idő többszörösére nő, és nem csak az aláírónak kell elvégeznie a számításokat, hanem a hitelesítő félnek is.
Adatmennyiség megnövekedése: az üzenettel együtt az aláírást is el kell küldeni, ami ebben az esetben méretben megegyező nagyságú.
Biztonsági limitek: Oszkár képes lehet átrendezni az üzenetek sorrendjét vagy új üzeneteket tud létrehozni korábbi üzenet és aláírás töredékekből. Habár a támadó nem tud blokkokon belül manipulálni, de a teljes üzenet biztonsága nem biztosított.
Éppen ezért szükségünk van egy rövid aláírásra egy adott hosszúságú üzenethez. Erre nyújt megoldást a hash függvény. Amennyiben van egy hatásos hash függvényünk, ami kiszámítja az üzenetünk egy lenyomatát, akkor alá tudjuk írni az üzenetet.
Alap protokoll a hash függvényt használó digitális aláírásra:

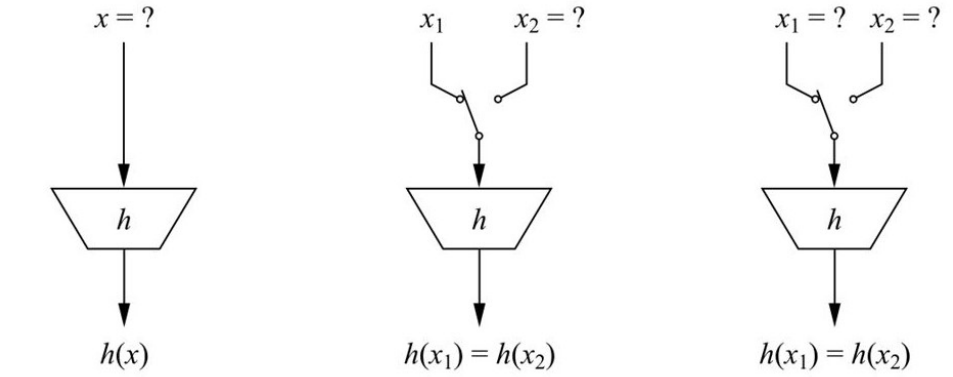
Aliz		**Bob**
	$\xleftarrow{k_{pub,B}}$	
		$z = h(x)$
		$s = sig_{k_{pr,B}}(z)$
	$\xleftarrow{(x,s)}$	
$z' = h(x)$		
$ver_{k_{pub,B}}(s, z')$ =true/false		

Bob kiszámítja az üzenet hash függvényét és aláírja a hash értéket: z-t a $k_{pr,B}$ titkos kulcsával. A fogadó oldalon Aliz kiszámítja az üzenet hash értékét: z'-t. Hitelesíti az aláírást: s-t, Bob publikus kulcsának segítségével. Fontos, hogy az aláírás generálást és a hitelesítést is a hashelt üzeneten számoljuk. A hash érték felel meg az üzenetnek ebben az esetben.
Amit szeretnénk elérni, hogy a hash függvényt bármilyen hosszú üzenetre tudjuk alkalmazni. Fontos, hogy gyorsan és hatékonyan tudjunk vele számolni, és adott hosszúságú legyen a függvény kimenete, függetlenül az eredeti üzenet hosszától. A hashalt üzenet hossza jó esetben 128-512 bit közé esik. Továbbá fontos, hogy az üzeneten történő legkisebb változtatás esetén is teljesen különböző lenyomatot kapjunk.

2.2.2. Biztonsági megkötések

Számos támadás épül arra, hogy a hash függvények gyengeségeit kihasználva támadnak egy protokollt. Éppen ezért egy biztonságosnak tekinthető hash függvénynek három fontos tulajdonsággal kell rendelkeznie:

- előkép ellenálló = egyirányú (preimage resistance)
- másod előkép ellenálló = gyengén ütközésmentes (second preimage resistance)
- ütközésmentes (collision resistance)



preimage resistance | second preimage resistance | collision resistance

[5]

Egyirányúság:
A hash függvények egyirányúsága azt jelenti, hogy a hash értékéből nem határozható meg egyértelműen az eredeti üzenet. Más szóval, figyelni kell arra, hogy a lenyomatból ne tudjuk visszaállítani az eredeti üzenetet.
Gyengén ütközésmentesség:
Digitális aláírásoknál fontos, hogy két különböző üzenetnek ne legyen ugyanaz a hashelt értéke. Két esetet különböztethetünk meg:
Az első eset, ha adott x_1 és x_2-őt szeretnénk megtalálni. A másik eset pedig, ha a támadónak kell megkeresnie x_1-et és x_2-őt is. Ezzel a következő részben foglalkozunk.

Aliz	**Oszkár**	**Bob**
		$\xleftarrow{k_{pub,B}}$
		$z = h(x_1)$
		$s = sig_{k_{pr,B}}(z)$
	$\xleftarrow{(x_2,s)}$ helyettesítés	$\xleftarrow{(x_1,s)}$
$z = h(x_2)$		
$ver_{k_{pub,B}}(s,z) = \text{true}$		

Tegyük fel, hogy Bob kiszámítja az x_1 üzenet hash értékét. Ha Oszkár tud olyan x_2 üzenetet találni/generálni, hogy $h(x_1) = h(x_2)$, akkor Aliz az Oszkár által generált üzenetet elfogadja. A kérdés, hogy hogyan akadályozzuk meg Oszkárt, hogy találjon egy megfelelő x_2-őt. Ehhez kell találnunk olyan hash függvényt, ami gyengén ütközésmentes. Ez a skatulyaelv elv miatt majdnemhogy lehetetlen. Mivel a hash függvény kimenete adott bithosszúságú, (legyen n hosszú), ezért mindössze 2^n lehetséges kimenetel létezik. Ugyanakkor a hash függvények bemeneti értékeinek száma végtelen, szóval több üzenetnek is azonos értéket kell felvennie.
Egy erős hash függvényt úgy kell megtervezni, hogy adott x_1-re ne tudjunk olyan x_2-őt találni, hogy a két üzenet hash értéke megegyezzen. Azonban Oszkár bármikor képes véletlenszerűen kiválasztani egy üzenetet és megnézni, hogy a hash értéke milyen értéket vesz fel, majd addig próbálkoznia, amíg nem talál megfelelő üzenetet. Ahhoz, hogy ezt a fajta támadást kikerüljük, legalább $n = 80$ bithosszúságú üzenetekkel kell dolgoznunk.
Ütközésmentesség és a szülinap paradoxon:
Akkor nevezzük ütközésmentesnek a hash függvényt, ha nem tudunk két különböző üzenetet találni ugyanazzal a hash értékkel. $(x_1 \neq x_2, h(x_1) = h(x_2))$

Aliz	**Oszkár**	**Bob**
		$\xleftarrow{k_{pub,B}}$
		$\xrightarrow{x_1}$
		$z = h(x_1)$
		$s = sig_{k_{pr,B}}(z)$
	$\xleftarrow{(x_2,s)}$ helyettesítés	$\xleftarrow{(x_1,s)}$
$z = h(x_2)$		
$ver_{k_{pub,B}}(s,z) = \text{true}$		

Oszkár két üzenetet generál, majd apró módosításokat végez rajtuk (szóköz beszúrása stb.), így az üzenet jelentése nem változik de a hash függvény minden lépésben más értéket vesz fel. Addig kell folytatni ezeket a módosításokat, amíg $h(x_1) = h(x_2)$ teljesül. Ekkor Oszkár rá tudja venni Alizt, hogy írja alá x_1-et. Ahogy korábban láttuk, a skatulyaelv miatt mindig létezik ütközés. A kérdés az, hogy mennyire nehéz megtalálni. Első tippre azt gondolhatjuk, hogy hasonlóan az előző példához, ha a kimenetel 80 bit hosszú, akkor 2^{80} üzenetet kell leellenőriznünk. Azonban a gyakorlatban elég 2^{40} üzenetet megvizsgálni. Ez a meglepő eredmény az úgynevezett születésnap támadás miatt lehetséges, ami a szülinap paradoxonra épül.

A következő valós probléma nagyon is kapcsolódik a hash függvények ütközésének megkereséséhez: Hány ember legyen ott egy partin, ha azt szeretnénk, hogy legyen köztük legalább két ember, akik egy napon születtek? Első megérzésre 183 embert tippelhetünk. Valójában sokkal kevesebb ember is elég. Számítsuk ki annak az esélyét, hogy két embernek nem ugyanazon a napon van a születésnapja. Egy emberre nézve ennek az esélye 1, ami triviális. Két emberre nézve ez már 364 alatt a 365.//

P(nincs ütközés két ember között)= $(1 - \frac{1}{365})$

Ha egy újabb ember érkezik a buliba, akkor mindkét már ottlevő emberrel lehet ütközés:

P(nincs ütközés három ember között)= $(1 - \frac{1}{365}) \cdot (1 - \frac{2}{365})$

Ennek következtében, annak az esélye, hogy t embernél nincs ütközés:

P(nincs ütközés t ember között)= $(1 - \frac{1}{365}) \cdot (1 - \frac{2}{365}) \cdots (1 - \frac{t-1}{365})$.

$t = 366$ emberre nézve a valószínűség 1 lesz, hiszen egy év csak 365 napból áll, így mindenképpen lesz két ember, akinek egy napon van a születésnapja. Visszatérve az eredeti kérdésünkre, hány ember kell ahhoz, hogy 50% eséllyel legyen

legalább két olyan ember, akinek egy napra esik a szülinapja. A fenti egyenlet alapján, meglepő módon 23 ember is elegendő ehhez:

$$\begin{aligned} &\mathrm{P}(\text{legalább egy ütközés}) = 1 - \mathrm{P}(\text{nincs ütközés}) \\ &= 1 - (1 - \frac{1}{365}) \cdots (1 - \frac{23-1}{365}) \\ &= 0.507 \approx 50\% \end{aligned}$$

Ütközés keresés a hash függvényeknél ugyanaz a probléma, mint születésnap ütközést keresni egy parti résztvevő között. A hash függvényeknél azonban nem 365 értéket vehet fel minden elem, hanem 2^n-t, ahol n a hashelt érték hossza és egyben a kritikus biztonsági paramétere is a függvénynek. A kérdés az, hogy hány üzenetet kell Oszkárnak hashelnie, amíg nem talál két olyan üzenetet, aminek egyezik a hashelt értéke. Annak az esélye, hogy nincs ütközés t hash érték között:

$$\mathrm{P}(\text{nincs ütközés}) = (1 - \tfrac{1}{2^n})(1 - \tfrac{2}{2^n}) \cdots (1 - \tfrac{t-1}{2^n}) = \prod_{i=1}^{t-1}(1 - \frac{i}{2^n})$$

Tudjuk, hogy $e^{-x} \approx 1 - x$, mivel $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots, x << 1$.
Mivel $i/2^n << 1$, ezért fel tudjuk írni annak az esélyét, hogy nincs ütközés, az alábbi formában:

$$\begin{aligned} &\approx \prod_{i=1}^{t-1} e^{-\frac{i}{2^n}} \\ &\approx e^{-\frac{1+2+3+\cdots+t-1}{2^n}} \\ &\approx e^{-\frac{t(t-1)}{2 \cdot 2^n}} \end{aligned}$$

Célunk, hogy megtaláljuk, hogy hány darab üzenet szükséges ahhoz, hogy ütközést találjunk. Ezért, ezt az egyenletet t-re rendezve kell megoldanunk. Jelöljük λ-val annak az esélyét, hogy legalább egy ütközés van: $\lambda = 1 - \mathrm{P}(\text{nincs ütközés})$.

$$\lambda \approx 1 - e^{-\frac{t(t-1)}{2 \cdot 2^n}}$$

$$\ln(1-\lambda) \approx -\frac{t(t-1)}{2 \cdot 2^n}$$

$$t(t-1) \approx 2^{n+1} \ln(\frac{1}{1-\lambda})$$

Mivel $t >> 1$, ezért $t^2 \approx t(t-1)$:

$$t \approx \sqrt{2^{n+1} \ln(\frac{1}{1-\lambda})}$$

$$t \approx 2^{(n+1)/2} \sqrt{\ln(\frac{1}{1-\lambda})}$$

8. Definíció (Biztonsági szint). *Azt mondjuk, hogy egy algoritmus biztonsági szintje n, ha a legjobb ismert támadásnak 2^n lépésre van szüksége a feltöréshez.*

A legfontosabb következménye a születésnap támadásnak, hogy azoknak az üzeneteknek a száma amit hashelnünk kell ahhoz, hogy találjunk ütközést, nagyjából a lehetséges kimeneti értékek számának négyzetgyökével egyenlő: $\sqrt{2^n} = 2^{n/2}$. Ezért ahhoz, hogy elérjünk egy x biztonsági szintet, a hash függvénynek a kimeneti hosszának legalább $2x$ hosszúnak kell lennie.

Példa

Tegyük fel, hogy 50% eséllyel szeretnénk ütközést találni egy hipotetikus hash függvénynél, aminek 80 bit hosszú a kimeneti értéke:

$t = 2^{81/2} \sqrt{\ln(1/(1-0.5))} \approx 2^{40.2}$

3. Algoritmusok diszkrét logaritmus számításához

3.1. Ciklikus csoporton értelmezett diszkrét logaritmus

A modern kriptográfiában a diszkrét logaritmus probléma [5] egyike a legfontosabb kriptográfiai szempontból egyirányú függvényeknek. Számos nyilvános kulcsú algoritmus épül rá. Adott $\mathbb{G}$ ciklikus csoport q elemszámmal, aminek $g \in \mathbb{G}$ a generátora: $\mathbb{G} = \{g^0, g^1, g^2, ..., g^{q-1}\}$. Azaz, minden $h \in \mathbb{G}$-re létezik egy egyedi $x \in \mathbb{Z}_q$, amire $q^x = h$. Ezt az x-et nevezzük a h g alapú diszkrét logaritmusának. Jelölés: $x = \log_g h$. Amennyiben $g^{x'} = h$ (ahol x' tetszőleges egész szám), akkor $\log_g h = [x' \bmod q]$. Ebben az esetben azért "diszkrét" logaritmusról beszélünk, mert az értékeket véges értékkészletből vesszük, ezzel szemben a "standard" logaritmus értékei egy végtelen halmazból kerülnek ki. A diszkrét logaritmusra hasonló szabályok vonatkoznak, mint a standard logaritmusra, ezek közül kettőt kiemelve:

- $\log_g 1 = 0$, ahol 1 az egységeleme $\mathbb{G}$-nek
- $\log_g(h_1 \cdot h_2) = [(\log_{h_1} + \log_{h_2}) \bmod q]$.

Ciklikus $\mathbb{G}$ csoporton értelmezett diszkrét logaritmus problémán a $\log_g h$ kiszámítását értjük, ahol g és $h \in \mathbb{G}$ adott.

Másképp fogalmazva, adott $g \in \mathbb{G}$ és $y \in \langle g \rangle$, találjunk olyan x-et, hogy $g^x = h$. A megoldást $\log_g h$-nal jelöljük és egyértelmű modulo $o(g)$ szerint. Előfordul, hogy g-re úgy hivatkozunk, mint a diszkrét logaritmus probléma alapjára.

3.2. A diszkrét logaritmus probléma

Két csoportra oszthatjuk a diszkrét logaritmus problémát támadó algoritmusokat: azok amik tetszőleges csoporton működnek, illetve amik valamilyen speciális tulajdonsággal rendelkező csoporton működnek. Az első kategóriába sorolható algoritmusoknál, ahol $\mathbb{G}$ nem ciklikus, gyakran magát a $\langle g \rangle$ ciklikus alcsoportot használjuk. Ekkor hagyjuk, hogy q határozza meg $\langle g \rangle$ rendjét és feltesszük, hogy q ismert.

A $\log_g y$ diszkrét logaritmusra alkalmazott brute-force keresés (teljes kipróbálás módszere) $\mathcal{O}(q)$ idő alatt fut le, ezért ennél gyorsabb algoritmusokkal érdemes foglalkozni.

A továbbiakban két algoritmust részletezek:

- Shank kis lépés/nagy lépés algoritmusa [3] , ami q rendű csoportokban számolja ki a diszkrét logaritmust $\mathcal{O}(\sqrt{q} \cdot polylog(q)$ idő alatt.

- Pohlig-Hellman algoritmus [3] amit akkor tudunk használni, ha ismert a csoport rendjének, q-nak a felbontása. Amennyiben q-nak "kis" tényezői vannak, a feladat több kisebb részfeladatra bontható és ezekre alkalmazva adott diszkrét logaritmust megoldó algoritmusokat, gyorsabb futási időt eredményezhet ($\mathcal{O}(polylog(q) \cdot max_i\{\sqrt{p_i}\})$).

3.3. "Kis lépés/nagy lépés" algoritmus

Shanks algoritmusa q rendű csoportban számolja ki a diszkrét algoritmust $\mathcal{O}(\sqrt{q} \cdot polylog(q))$ időben. Az ötlet igencsak egyszerű. Adott a q bemenet és $y \in \langle g \rangle$. $\langle g \rangle$ elemeit egy körként kell elképzelni és tudjuk, hogy y valahol ezen a körön "fekszik". $1 = g^0, g^1, g^2, ...g^{q-1}, g^{q-2}, g^q = 1$ A kör összes pontjának kiszámítása és leírása $\Omega(q)$ időt venne igénybe. Ehelyett $t \overset{def.}{=} \lfloor\sqrt{q}\rfloor$ méretű részekre (intervallumokra) bontjuk a kört, majd kiszámítjuk és feljegyezzük a $g^0, g^t, g^{2t}...g^{\lfloor q/t \rfloor}$ elemeket. Ezek az úgynevezett "nagy lépések".
A körön feltüntetett egymást követő szakaszok elemszáma legfeljebb t. Továbbá tudjuk, hogy $y = g^x$ a szakaszok egyikén található. Ezért biztosan tudjuk, hogy az általunk feljegyzett elemek egyike, meg fog egyezni az $y \cdot g^0 = g^x, y \cdot g^1 = g^{x+1}, ..., y \cdot g^t = g^{x+t}$, elemek egyikével. (Kis lépések)
Legyen $y \cdot g^i = g^{k \cdot t}$. Innen már könnyen számolható: $y = g^{k \cdot t - i}$ vagy $\log_g y = [k \cdot t - i \bmod q]$.
Az algoritmus pszeudokódja:

Be: $g \in \mathbb{G}$ elemei és $y \in \langle g \rangle$, g rendje: q
Ki: $\log_g y$
$t := \lfloor\sqrt{q}\rfloor$
for $i = 0$ to $\lfloor q/t \rfloor$:
$g_i := g^{i \cdot t}$
(i, g_i) párok rendezése a második komponens alapján
for $j = 0$ to t:
$y_j := y \cdot g^j$
$if y_j = g_k$ valamilyen k-ra, return $[k \cdot t - j \bmod q]$

Az algoritmus $\mathcal{O}(\sqrt{q})$ hatványozást és szorzást igényel $\mathbb{G}$-ben és mindegyik hatványozás elvégezhető $\mathcal{O}(polylog(g))$ idő alatt egy hatásos hatványozó algoritmussal. g_1-en és g_t-n kívül ($g_1 = g_t$) minden más g_i kiszámolható egy szorzással: $g_i := g_{i-1} \cdot g_1$. Hasonlóan: $y_i := y_{i-1} \cdot g$. Az (i, g_i) párok rendezése $\mathcal{O}(\sqrt{q} \cdot \log_q)$ időt vesz igénybe és utána bináris kereséssel ellenőrizhetjük, hogy y_i egyenlő-e valamelyik g_k-val: $\mathcal{O}(\log_q)$. Ezeket figyelembe véve kijön, hogy az algoritmus

valóban $\mathcal{O}(\sqrt{q} \cdot polylog(q))$ idő alatt fut.

Példa:

Legyen $\mathbb{Z}_{29}^*$, $q = 29 - 1 = 28$, $g = 2$, $y = 17$, t-nek pedig válasszuk a $t = 5$-öt.
$2^0 = 1$, $2^5 = 3$, $2^{10} = 9$, $2^{15} = 27$, $2^{20} = 23$, $2^{25} = 11$
$17 \cdot 2^0 = 17$, $17 \cdot 2^1 = 5$, $17 \cdot 2^2 = 10$, $17 \cdot 2^3 = 20$, $17 \cdot 2^4 = 11$, $17 \cdot 2^5 = 22$
Vegyük észre, hogy $2^{25} = 11 = 17 \cdot 2^4$. Innen: $\log_2 17 = 25 - 4 = 21$.

3.4. A Pohling-Hellman algoritmus

A Pohling-Hellman algoritmussal felgyorsíthatjuk a számításokat, amennyiben q-nak (csoport rendje) ismert egy nem-triviális tényezője. A g rendje $\mathsf{ord}(g)$, a legkisebb pozitív i, amire $g^i = 1$.

Lemma: Legyen $\mathsf{ord}(g) = q$ és tegyük fel, hogy $p|q$. Ekkor $\mathsf{ord}(g^p) = q/p$.
Bizonyítás: Mivel $(g^p)^{q/p} = g^q = 1$, g^p rendje legfeljebb q/p. Legyen $i > 0$, hogy $(g^p)^i = 1$. Ekkor $g^{p \cdot i} = 1$ és mivel g rendje q, ezért $p \cdot i \geqslant q$ és $q/p \geqslant i$. g^p rendje ezért pontosan q/p.

Algoritmus: Legyenek adottak g és y, keressük x-et, hogy $g^x = y$. Legyen $\mathsf{ord}(g) = q$ és ismert $q = \prod_{i=1}^{k} q_i$, ahol q_i-k páronként relatív prímek. Tudjuk, hogy $(g^{q/q_i})^x = (g^x)^{q/q_i} = y^{q/q_i}$ $i = 1, ..., k$. Legyen $g_i \overset{def.}{=} g^{q/q_i}$, így kapunk k darab kisebb megoldandó problémát, mindegyik $\mathsf{ord}(g_i) = q_i$ mérettel.
Mindegyik kisebb részproblémát megoldhatjuk különböző algoritmusokkal. Konkrét példát felhozva: tegyük fel, hogy az előző példában használt kis lépés/nagy lépés algoritmust használjuk. A kisebb részek megoldásaként kapjuk az $\{x_i\}_{i=1}^k$ megoldásokat, amire $g_i^{x \cdot i} = y^{q/q_i} = g_i^x$.
$x = x_i \bmod q_i$ minden i-re. A kínai maradék tétel felhasználásával:

$$\begin{aligned} x &\equiv x_1 \mod q_1 \\ x &\equiv x_2 \mod q_2 \\ &\vdots \\ x &\equiv x_k \mod q_k \end{aligned}$$

. A részfeladatok egyértelműen meghatározzák x-et $\bmod q$. Ezért x hatékonyan számolható x_1, x_2,...,x_k-ból.

Példa:
$\mathbb{Z}_p^*$-n legyen $p = 31$, $\mathbb{Z}_{31}^*$, $q = 31 - 1 = 30 = 5 \cdot 3 \cdot 2$, $g = 3$, $y = 26 = g^x$.

$$(g^{30/5})^x = y^{30/5} \Rightarrow (3^6)^x = 26^6 \Rightarrow 16^x = 1 \bmod 31$$

$$(g^{30/3})^x = y^{30/3} \Rightarrow (3^10)^x = 26^10 \Rightarrow 25^x = 5 \bmod 31$$

$$(g^{30/2})^x = y^{30/2} \Rightarrow (3^15)^x = 26^15 \Rightarrow 30^x = 30 \bmod 31$$

Megoldva az egyenlőségeket:

$$x = 0 \bmod 5$$

$$x = 2 \bmod 3$$

$$x = 1 \bmod 2$$

$$\Rightarrow x = 5 \bmod 30, 3^5 = 26 \bmod 31.$$

Feltételezve, hogy ismert q egy jó faktorizálása, és a Kis lépés/nagy lépés algoritmust használva a kisebb részfeladatokra az algoritmus futási ideje: $\mathcal{O}(polylog(q) \cdot \sum_{j=1}^{k} \sqrt{q_i})$. Mivel q-nak legfeljebb $\log q$ tényezője lehet, a képlet leegyszerűsödik $\mathcal{O}(polylog(q) \cdot max_i\{\sqrt{q_i}\})$.
Attól függően, hogy mekkora a q-nak a legnagyobb ismert tényezője, jelentős javítást érhetünk el az előbb megismert $\mathcal{O}(\sqrt{q})$-hoz képest. Ha olyan felbontását ismerjük, ami sok kicsi tényezőből áll, akkor a diszkrét logaritmus probléma q rendű csoportban számolva relatív könnyen törhető.
Ha q faktorizálása: $q = \prod_{i=1}^{k} p_i^{e_i}$, akkor a Pohling-Hellman algoritmus a fentiek alapján $\mathcal{O}(polylog(q) \cdot max_i\{\sqrt{p_i^{e_i}}\})$ idő alatt fut le. További javításokat hozzávéve az algoritmus $\mathcal{O}(polylog(q) \cdot max_i\{\sqrt{p_i}\})$ idő alatt is futhat.

4. Diffie-Hellman Kulcscsere

4.1. A Diffie-Hellman protokoll

A Diffie-Hellman kulcscserét [5] Whitfield Diffie és Martin Hellman mutatta be 1976-ban, ez volt az első publikált asszimetrikus séma. Praktikus megoldást nyújt a kulcsmegosztás problémára, például lehetővé teszi két fél számára, hogy egy közös titkos kulcsot hozzanak létre egy nem biztonságos csatornán. A protokollt számos nyílt forráskódú és kereskedelmi kriptográfiai protokollnál használják, mint például a Secure Shellnél, Transport Layer Security-nél és az Internet Protocol Security-nél. A Diffie-Hellman kulcscsere alap ötlete az volt, hogy a hatványozás egy egyirányú függvény és kommutatív $\mathbb{Z}_p^*$-n, ahol p prím. Legyen $k = (\alpha^x)^y \equiv (\alpha^y)^x \mod p$. A $k \equiv (\alpha^x)^y \equiv (\alpha^y)^x \mod p$ a közös titok,amit a folyamat kulcsaként használhatunk a két fél között.
A Diffie-Hellman kulcscserénél két résztvevőnk van: Aliz és Bob, akik szeretnének létrehozni egy közös titkos kulcsot. Lehetséges, hogy legyen egy megbízható harmadik fél is, aki kiválasztja a kulcscseréhez szükséges publikus paramétereket, ez azonban nem szükséges. A kulcscsere két protokollból áll, egy előkészületből és magából a kulcscseréből. Az előkészület a következő lépésekből tevődik össze:

- Elég nagy p prím megválasztása
- $\alpha \in \{2, 3, ..., p-2\}$ megválasztása
- p és α publikálása

Amennyiben Aliz és Bob is ismerik a két paramétert, akkor létre tudnak hozni egy k közös titkos kulcsot.
A kulcscsere menete:

Aliz		**Bob**
$a = k_{pr,A} \in \{2, ..., p-2\}$		$b = k_{pr,B} \in \{2, ..., p-2\}$
$A = k_{pub,A} \equiv \alpha^a \mod p$		$B = k_{pub,B} \equiv \alpha^b \mod p$
	$\xrightarrow{k_{pub,A}=A}$	
	$\xleftarrow{k_{pub,B}=B}$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \mod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \mod p$

Aliz választ egy $a = k_{pr,A} \in \{2, ..., p-2\}$ számot és kiszámolja $A = k_{pub,A} \equiv \alpha^a \mod p$-ot, ezt küldi tovább Bobnak. Bob Alizhoz hasonlóan kiválaszt egy b számot és kiszámítja B-t és továbbküldi Aliznak:$B = k_{pub,B} \equiv \alpha^b \mod p$. Innen Aliz és Bob is ki tudja számítani $k_{A,B}$-t: $k_{pub,B}^{k_{pr,A}} \equiv B^a \mod p$, illetve:$k_{pub,A}^{k_{pr,B}} \equiv A^b \mod p$.

Könnyen megbizonyosodhatunk arról, hogy Aliz és Bob valóban ugyanazt a k_{AB} kulcsot számítja ki:
Aliz: $B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \mod p$
Bob: $A^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \mod p$.

Példa a DHKE-re:

Legyen $p = 29$ és $\alpha = 2$.

Aliz		**Bob**
$a = k_{pr,A} = 5$		$b = k_{pr,B} = 12$
$A = k_{pub,A} = 2^5 \equiv 3 \mod 29$		$B = k_{pub,B} = 2^{12} \equiv 7 \mod 29$
	$\xrightarrow{A=3}$	
	$\xleftarrow{B=7}$	
$k_{AB} = B^a = 7^5 \equiv 16 \mod 29$		$k_{AB} = A^b = 3^{12} \equiv 16 \mod 29$

4.2. Biztonságos kulcscsere

Azok a protokollok, amik a DHKE-t használják, védtelenek az aktív támadásokkal szemben. Ez annyit tesz, hogy ha van egy harmadik kívülálló fél is a kulcscserénél (Oszkár), aki képes megváltoztatni az üzenetet vagy létrehozni nem valósat, akkor Oszkár fel tudja törni az üzeneteket. Ez az úgynevezett "man-in-the-middle" támadás.
Tegyük fel, hogy Oszkár egy passzív támadó, nem tudja módosítani az üzeneteket, csak lehallgatni. Célja, hogy az Aliz és Bob által megosztott k_{AB} kulcsot ő is ki tudja számítani. Oszkár a következő információkhoz tud jutni, azzal hogy lehallgatja a beszélgetést: α-t és p-t (mindkettő publikus), és ha figyeli a kommunikációs csatornát a kulcscsere folyamatánál, akkor A-t és B-t is le tudja hallgatni. Innen már csak az a kérdés, hogy a megszerzett információból képes-e Oszkár kiszámolni $k = \alpha^{ab}$-t a következő értékekből: α, p, $A \equiv \alpha^a \mod p$, $B \equiv \alpha^b \mod p$? Ezt a problémát nevezzük Diffie-Hellman problémának (DHP). Ugyanúgy, mint a diszkrét logaritmus probléma, ez is általánosítható tetszőleges, véges ciklikus csoportokra.

9. Definíció (Általánosított Diffi-Hellman számítási probléma). *Legyen adott véges, n-ed rendű ciklikus csoport: $\mathbb{G}$, a primitív eleme: $\alpha \in \mathbb{G}$ és két eleme: $A = \alpha^a \in \mathbb{G}$ és $B = \alpha^b \in \mathbb{G}$. A Diffi-Hellman számítási probléma: megtalálni az α^{ab} csoport elemet (DHP).*

A DHKE törése $\mathbb{Z}_p^*$-n: tegyük fel, hogy Oszkár tud egy hatékony módszert a diszkrét logaritmus problémára az adott csoportban. Ekkor a DHP-t meg tudja oldani és és ki tudja számítani k_{AB}-t a következő módon:

- Oszkár kiszámítja Aliz privát kulcsát ($a = k_{pr,A}$) úgy, hogy megoldja az alábbi diszkrét logaritmust: $a \equiv \log_\alpha a \mod p$.
- Megoldja: $k_{AB} \equiv B^a \mod p$.

Bár könnyűnek tűnhet, de ha a p prím elég nagy, akkot a diszkrét logaritmust nem lehet kiszámítani.
Az még nem bizonyított, hogy ez lenne az egyetlen módja annak, hogy feltörjék az DHKE-t, elméletben ugyanis lehetséges, hogy diszkrét logaritmus számítás nélkül is törhető legyen a protokoll. Viszont annak ellenére, hogy matematikai értelemben nincs bizonyítva, mégis úgy vesszük a gyakorlatban, hogy a DLP megoldása az egyetlen járható út a DHP feltöréséhez.
Annak érdekében, hogy a gyakorlatban is biztonságosan használható legyen a DHKE, meg kell bizonyosodnunk arról, hogy a DLP-t nem lehet megoldani. Ehhez szükséges, hogy megfelelően nagy p prímet válasszunk.
A 80-as biztonsági szint eléréséhez 1024 bites prímet kell választanunk, míg a 128 bites biztonsághoz már 3072 bites prímre van szükségünk. Továbbá, a Pohling-Hellman támadás megakadályozásához kell, hogy a csoport rendjének $(p-1)$ tényezőkre bontásánál, a felbontás nem állhat csak kis tényezőkből, ezzel megakadályozva a részproblémák létrehozását. Továbbá a legkisebb tényezőnek legalább 160 bit hosszúnak kell lennie ahhoz, hogy elérhető legyen a 80 bites biztonsági szint és 256 bit hosszúnak, ha a 128 bites szintet akarjuk elérni.

5. Elgamal titkosítási séma

5.1. Diffie-Hellman kulcscseréből Elgamal titkosítás

Az Elgamal sémát Taher Elgamal dolgozta ki 1985-ben [5] . A DHKE protokoll kiterjesztéseként is ismert. A DHKE-hez hasonlóan az Elgamal biztonsága is a diszkrét logaritmus és a Diffie-Hellman probléma gyakorlati visszafejthetetlenségén alapszik. Az Elgamal sémát $\mathbb{Z}_p^*$-n értelmezzük, ahol p prím. Más ciklikus csoportokban is értelmezhetjük, amelyekben a DL és DH probléma is nehezen megoldható, például a Galois mezőn $GF(2^m)$.

Az Elgamal séma a DHKE problémára épül: itt is van két fél aki kommunikálni szeretne: Aliz és Bob. Ha Aliz egy titkosított x üzenetet szeretne küldeni Bobnak, akkor először a Diffie-Hellman kulcscsere segítségével létrehoznak egy k_M kulcsot. A kulcscserénél feltesszük, hogy létrehoztak egy elég nagy p prímet és egy α primitív elemet. Ezután Aliz ezt a kulcsot egy multiplikatív maszknak használva titkosítja az x üzenetet: $y \equiv x \cdot k_M \mod p$.
A protokoll tehát két részből áll: a DHKE-ből (a-f), valamint az azt követő titkosításból és dekódolásból (g,h).

Aliz		**Bob**
		(a) $d = k_{pr,B} \in \{2, ..., p-2\}$
		(b) $\beta = k_{pub,B} \equiv \alpha^d \mod p$
	$\xleftarrow{\beta}$	
(c) $i = k_{pr,A} \in \{2, ..., p-2\}$		
(d) $k_E = k_{pub,A} \equiv \alpha^i \mod p$		
	$\xrightarrow{k_E}$	
(e) $k_M \equiv \beta^i \mod p$		(f) $k_M \equiv k_E^d \mod p$
(g) $x \in \mathbb{Z}_p^*$ titkosítása:		
$y \equiv x \cdot k_M \mod p$		
	$\xrightarrow{y}$	
		(h) dekódolás: $x \equiv y \cdot k_M^{-1} \mod p$

Bob kiszámítja a d privát kulcsát és β publikus kulcsát. Ez a kulcspár nem változik, sok üzenetet lehet vele titkosítani. Aliznak azonban minden egyes új üzenetnél ki kell számítania a nyilvános (k_E) és titkos (i) kulcsokat. A titkosításnál Aliz összeszorozza az x üzenetet a maszkoló kulccsal (k_M) $\mathbb{Z}_p^*$-n. Az üzenetfogadó pedig az inverz maszkkal való szorzással oldja fel az üzenetet.

5.2. Az Elgamal protokoll

Formálisabban is leírható a séma, ekkor három részre bontjuk fel a folyamatot. Az első fázist az üzenetfogadó hajtja végre, az ő feladata a publikus kulcs meghatározása. A titkosítás és az üzenet feloldása is egy-egy részt képez, és minden egyes alkalommal, amikor üzenet érkezik, végre kell hajtani őket. A DHKE-vel ellentétben itt nincs szükség egy megbízható harmadik félre. Az üzenetfogadó választja ki a prímet és a primitív elemet is ő teszi nyilvánossá őket.

Aliz		**Bob**
		$p,\ \alpha \in \mathbb{Z}_p^*$
		$k_{pr} = d \in \{2, ..., p-2\}$
		$k_{pub} = \beta = \alpha^d \mod p$
	$\xleftarrow{k_{pub}=(p,\alpha,\beta)}$	
$i \in \{2, ..., p-2\}$		
$k_E \equiv \alpha^i \mod p$		
$k_M \equiv \beta^i \mod p$		
$x \in \mathbb{Z}_p^*$ üzenet titkosítása:		
$y \equiv x \cdot k_M \mod p$		
	$\xrightarrow{(k_E,y)}$	
		$k_M \equiv k_E^d \mod p$
		$x \equiv y \cdot k_M^{-1} \mod p$

Ennél a módszernél felcseréljük a műveletek sorrendjét, így Aliznek elegendő csupán egy üzenetet küldenie Bobnak, az előző protokollban látható két üzenet helyett.

A titkosított üzenet két részből áll. Az ideiglenes kulcsból, k_E-ből és a maszkolt üzenetből, y-ból. Mivel általában az összes paraméter $\lceil \log_2 p \rceil$ bit hosszú, ezért a (k_E, y) titkosított üzenet ennek a kétszerese.

A protokoll helyességét az alábbi módon ellenőrizhetjük:

$$\begin{aligned} d_{k_{pr}}(k_E, y) &\equiv y \cdot (k_M)^{-1} \mod p \\ &\equiv [x \cdot k_M] \cdot (k_E^d)^{-1} \mod p \\ &\equiv [x \cdot (\alpha^d)^i][(\alpha^i)^d]^{-1} \mod p \\ &\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \equiv x \mod p \end{aligned}$$

Példa kis számokkal:
Bob generálja a kulcsokat és Aliz titkosítja az $x = 26$ üzenetet.

Aliz		**Bob**
$x = 26$		$p = 29$, $\alpha = 2$
		$k_{pr,B} = d = 12$
		$\beta = \alpha^d \equiv 7 \mod 29$
	$\xleftarrow{k_{pub,B}=(p,\alpha,\beta)}$	
$i = 5$		
$k_E = \alpha^i \equiv 3 \mod 29$		
$k_M = \beta^i \equiv 16 \mod 29$		
$y = x \cdot k_M \equiv 10 \mod 29$		
	$\xrightarrow{(k_E,y)}$	
		$k_M = k_E^d \equiv 16 \mod 29$
		$x = y \cdot k_M^{-1} \equiv 10 \cdot 20 \equiv 26 \mod 29$

Fontos megjegyezni, hogy i véletlenszerű megválasztása miatt $\{2, 3, ..., p-2\}$-ből, ha két különböző üzenetet szeretnénk titkosítani ($x_1, x_2 \in \mathbb{Z}_p^*$) ugyanazt a publikus kulcsot használva, akkor nagy valószínűséggel két különböző titkosított üzenetet fogunk kapni ($y_1 \neq y_2$). Így a brute-force keresés x-re eleve kivédhető.

5.3. Biztonság

Amennyiben az Elgamal titkosítás biztonságát szeretnénk elemezni, fontos elkülönítenünk az aktív és passzív támadásokat egymástól.

Passzív támadásnál a harmadik fél visszafejti az x üzenetet $p, \alpha, \beta, k_E = \alpha^i$ és $y = x \cdot \beta^i$-ből. Jelenleg nincs más ismert módszer a DHP probléma megoldására,mint a diszkrét logarimus kiszámítása. Ha feltesszük, hogy Oszkár birtokában van olyan

képességeknek, aminek a segítségével tud DLP-t számolni, akkor két féle módon tudja támadni az Elgamal sémát.
Az első módszernél Bob d titkos kulcsát kell megfejtenie:

$$d = \log_\alpha k \mod p.$$

Ez a lépés megoldja a DLP-t, viszont ha a paraméterek jól vannak megválasztva, akkor nem lehet megoldani polinomiális időben. Amennyiben mégis sikerül Oszkárnak a számítás, akkor fel tudja oldani a szöveg titkosítását:

$$x \equiv y \cdot (k_E^d)^{-1} \mod p.$$

A második módszernél, ahelyett, hogy Bob titkos exponensét, d-t számítaná ki, Oszkár megkísérli Aliz random i exponensét kiszámítani:

$$i = \log_\alpha k \mod p.$$

Ugyanúgy, mint az előző módszernél, itt is egy DLP-t kell megoldani. Amennyiben sikerül Oszkárnak, ki tudja számolni x-et:

$$x \equiv y \cdot (\beta^i)^-1 \mod p.$$

Oszkárnak mindkét esetben meg kell oldania a DLP-t a véges $\mathbb{Z}_p^*$ csoporton, így p-nek mindenképpen legalább 1024 bit hosszúnak kell lennie.

Aktív támadás esetén meg kell bizonyosodnunk arról, hogy a nyilvános kulcs hiteles. Ez annyit tesz, hogy a küldőnek azzal a kulccsal kell rendelkeznie, ami a fogadótól származik. Amennyiben a támadónak sikerül meggyőznie a küldőt, hogy az ő kulcsa a nyilvános kulcs, akkor könnyen fel tudja törni az üzenetet.
Ha a küldő fél többször használja ugyanazt a titkos i kulcsot, akkor Oszkár fel tudja törni az üzenetet. Tegyük fel, hogy Aliz két különböző üzenet elküldéséhez ugyanazt a titkos kulcsot használja. Ebben az esetben a két maszkoló kulcs ($k_M = \beta^i$) is meg fog egyezni, és az ideiglenes kulcsok is azonos értéket fognak felvenni. Így Aliz a következő két titkosított üzenetet küldi el a csatornán: (y_1, k_E) és (y_2, k_E). Ha a támadó ki tudja találni az első üzenetet, akkor ki tudja számítani a maszkoló kulcsot $k_M \equiv y_1 x_1^{-1} \mod p$. Ennek a segítségével ki tudja számítani x_2-őt:

$$x_2 \equiv y_2 k_M^{-1} \mod p.$$

Minden más üzenetet, ami ugyanazzal az i értékkel lett titkosítva, vissza lehet fejteni ezzel a módszerrel. Ahhoz, hogy ezt a támadást kivédjük, meg kell bizonyosodnunk arról, hogy az i kitevő nem ismétlődik.

Egy másik támadás, ha a támadó megvizsgálja a titkosított üzenetet (k_E, y) és kicseréli a következőre: (k_E, sy), ahol s egy szabadon választott egész szám. Ekkor az alábbiakat számolja ki:

$$\begin{aligned} d_{k_{pr}}(k_E, sy) &\equiv sy \cdot k_M^{-1} \mod p \\ &\equiv s(x \cdot k_M) \cdot k_M^{-1} \mod p \\ &\equiv sx \mod p. \end{aligned}$$

Ekkor a feloldott üzenet s többszöröse. A támadó nem képes feloldani a titkosított szöveget, de manipulálni tudja azt. A gyakorlatban nem használjuk gyakran az Elgamal sémát, de a hasonló támadások elkerülése miatt érdemes megismerkedni vele.

6. Elgamal digitális aláírási séma

6.1. Kulcs generálás

Az Elgamal digitális aláírási séma [5] eltér az azonos néven futó titkosító sémától. Mint minden publikus kulcsú sémánál, itt is van egy előkészület, ahol kiszámítjuk a kulcsokat. Azzal kezdünk, hogy keresünk egy nagy p prímet és megalkotunk egy DLP-t:

- p prím megválasztása
- $\alpha \in \mathbb{Z}_p^*$ megválasztása
- véletlenszerű $d \in \{2, 3, ..., p-2\}$ megválasztása
- $\beta = \alpha^d \mod p$ kiszámítása
- $k_{pub} = (p, \alpha, \beta), k_{pr} = d$

6.2. Aláírás és igazolás

A privát kulcs és a publikus kulcs paramétereinek felhasználásával az x üzenet aláírását az aláíró résznél számítjuk ki: $sig_{k_{pr}}(x, k_E) = (r, s)$. Az aláírás két egészből, r-ből és s-ből tevődik össze. Az aláíró folyamat két fő részből áll: kiválasztunk egy véletlenszerű k_E értéket, ami az ideiglenes privát kulcs lesz és ezzel számítjuk ki az x-hez tartozó aláírást. Először megválasztjuk $k_E \in \{0, 1, 2, ..., p-2\}$-t úgy, hogy $lnko(k_E, p-1) = 1$ teljesüljön, majd kiszámítjuk az aláíráshoz szükséges paramétereket:

$$r \equiv \alpha^{k_E} \mod p$$
$$s \equiv (x - d \cdot r) k_E^{-1} \mod p - 1$$

A fogadó oldalon pedig ellenőrizzük az aláírás helyességét:

$$t \equiv \beta^r \cdot r^s \mod p$$
$$\text{ha } t \equiv \alpha^x \mod p, \text{ akkor az aláírás érvényes}$$
$$\text{ha } t \not\equiv \alpha^x \mod p, \text{ akkor az aláírás érvénytelen}$$

Röviden megfogalmazva, a fogadó elfogadja az aláírást, ha a $\beta^r \cdot r^s \equiv \alpha^x \mod p$ egyenlőség teljesül, különben a hitelesítés elbukik.
Az Elgamal aláírási séma helyességének ellenőrzése:

$$\begin{aligned} \beta^r \cdot r^s &\equiv (\alpha^d)^r (\alpha^{k_E})^s \mod p \\ &\equiv \alpha^{dr + k_E s} \mod p. \end{aligned}$$

A kifejezés akkor tekinthető helyesnek, ha α^x-el egyenlő modulo p-vel.

$$\alpha^x \equiv \alpha^{dr+k_E s} \mod p$$

A Kis Fermat tétel szerint az egyenlőség akkor teljesül, ha az egyenlet mindkét oldalán lévő kitevő egyenlő modulo $p-1$-el és $(\alpha, p)= 1$:

$$x \equiv dr + k_E s \mod p-1$$

Ezt az egyenletet átalakítva kapjuk meg az egyenlőtlenséget s-re. Az lnko$(k_E, p-1) = 1$ feltételnek itt van jelentősége, mert az ideiglenes k_E kulcsot invertálnunk kell:

$$s \equiv (x - d \cdot r)k_E^{-1} \mod p-1.$$

Példa kis számokkal:

Aliz | **Bob**

Bob

$p = 29$, $\alpha = 2$, $d = 12$

$\beta = \alpha^d \equiv 7 \mod 29$

$\xleftarrow{(p,\alpha,\beta)=(29,2,7)}$

Aláírás az $x = 26$ üzenetre:

$k_E = 5$, $lnko(5, 28) = 1$

$r = \alpha^{k_E} \equiv 2^5 \equiv 3 \mod 29$

$s = (x - dr)k_E^{-1} \equiv (-10) \cdot 17 \equiv 26 \mod 28$

$\xleftarrow{(x,(r,s))=(26,(3,26))}$

$t = \beta^r \cdot r^s \equiv 7^3 \cdot 3^2 6 = 22 \mod 29$

$\alpha^x \equiv 2^2 6 \equiv 22 \mod 29$

$t \equiv \alpha^x \mod 29 \Rightarrow$ hiteles aláírás.

6.3. Biztonság

Az aláíró sémának a biztonsága a DLP-n alapszik. Ha Oszkár képes diszkrét logaritmust számolni, akkor ki tudja számítani a d titkos kulcsot β-ból és a k_E ideiglenes kulcsot r-ből. Ezek segítségével már alá tudja írni a tetszőleges üzeneteket az aláíró helyett. Éppen ezért a paramétereket úgy kell megválasztani, hogy a DLP-t hosszú idő alatt lehessen megoldani. Az egyik legfontosabb megkötés, hogy a p prímnek legalább 1024 bit hosszúnak kell lennie. Természetesen növelhetjük a biztonságot, ha nagyobb prímmel számolunk. Arra is ügyelni kell, hogy ki tudjuk

védeni a kis alcsoportokra vonatkozó támadásokat. Ennek elkerülésének érdekében a gyakorlatban az α primitív elem segítségével kell létrehozni az elsőrendű csoportot. Az ilyen csoportokban minden elem primitív és nem léteznek kis részcsoportok.

Az ideiglenes kulcs ismételt felhasználása

Amennyiben az aláíró többször használja ugyanazt az ideiglenes kulcsot, a támadó könnyedén ki tudja számítani az α titkos kulcsot. Oszkár megvizsgálja a két digitális aláírást és üzenetet $(x,(r,s))$. Oszkár észreveszi ha a két üzenet (x_1,x_2)ugyanazt az ideiglenes kulcsot használja (k_E), mivel az r értéke mindkét esetben ugyanaz lesz $(r_1 = r_2 = \alpha^{k_E})$. A két s érték különböző lesz, amire a következő egyenleteket tudjuk felírni:

$$s_1 \equiv (x_1 - dr)k_E^{-1} \mod p-1$$
$$s_2 \equiv (x_2 - dr)k_E^{-1} \mod p-1$$

Ez az egyenletrendszer csupán két ismeretlent tartalmaz: k_E ideiglenes kulcsot és Bob privát kulcsát d-t. A két egyenletet kivonva egymásból, majd átrendezve a következőket kapjuk:

$$s_1 - s_2 \equiv (x_1 - x_2)k_E^{-1} \mod p-1$$
$$k_E \equiv \frac{x_1 - x_2}{s_1 - s_2} \mod p-1$$

Amennyiben $\text{lnko}(s_1 - s_2, p-1) \neq 1$, akkor k_E több értéket is felvehet és Oszkárnak ki kell találnia, hogy melyiket használták az aktuális aláírási sémában. A k_E birtokában, Oszkár ki tudja számítani a d titkos kulcsot:

$$d \equiv \frac{x_1 - s_1 k_E}{r} \mod p-1.$$

A d titkos kulcs és a nyilvános paraméterek ismeretében, Oszkár könnyedén tud Bob helyett üzeneteket aláírni. Ennek elkerülése érdekében minden egyes új üzenet aláírásánál egy véletlenszám generátor által létrehozott ideiglenes kulcsot kell választanunk.

Példa kis számokkal:

Tegyük fel, hogy Oszkár lehallgat két üzenetet amiket Bob aláírt és ugyanazt az ideiglenes kulcsot használta fel mindkét üzenetnél.

1. $(x_1, (r, s_1)) = (26, (3, 26))$
2. $(x_2, (r, s_2)) = (13, (3, 1))$

Oszkár tudja Bob nyilvános kulcsát: $(p, \alpha, \beta) = (29, 2, 7)$. Az összegyűjtött információkkal Oszkár ki tudja számítani a többször felhasznált ideiglenes kulcsot:

$$\begin{aligned} k_E &\equiv \frac{x_1 - x_2}{s_1 - s_2} \mod p-1 \\ &\equiv \frac{26-13}{26-1} \equiv 13 \cdot 9 \\ &\equiv 5 \mod 28 \end{aligned}$$

Ezt felhasználva tudjuk kiszámítani d értékét:

$$\begin{aligned} d &\equiv \frac{x_1 - s_1 \cdot k_E}{r} \mod p-1 \\ &\equiv \frac{26 - 26 \cdot 5}{3} \equiv 8 \cdot 19 \\ &\equiv 12 \mod 28 \end{aligned}$$

Egzisztenciális hamisítás:

Lehetséges, hogy a támadó aláírást generáljon egy random x üzenethez. A támadónak meg kell győznie Alizt, hogy ő Bob. A támadás a következőképpen zajlik:

Aliz	**Oszkár**	**Bob**
		$k_{pr} = d$
		$k_{pub} = (p, \alpha, \beta)$
	$\xleftarrow{(p,\alpha,\beta)}$	$\xleftarrow{(p,\alpha,\beta)}$
	i, j egészek kiválasztása:	
	ahol lnko$(j, p-1) = 1$	
	Aláírás:	
	$r \equiv \alpha^i \beta^j \mod p$	
	$s \equiv -rj^{-1} \mod p-1$	
	Üzenet kiszámítása:	
	$x \equiv si \mod p-1$	
	$\xleftarrow{(x,(r,s))}$	
Hitelesítés:		
$t \equiv \beta^r \cdot r^s \mod p$		
mivel $t \equiv \alpha^x \mod p \Rightarrow$		
Érvényes aláírás		

Könnyen ellenőrizhető, hogy Aliz miért fogadja el Oszkár üzenetét:

$$\begin{aligned} t &\equiv \beta^r \cdot r^s \mod p \\ &\equiv \alpha^{dr} \cdot r^s \mod p \\ &\equiv \alpha^{dr} \cdot \alpha^{(i+dj)^s} \mod p \\ &\equiv \alpha^{dr} \cdot \alpha^{(i+dj)(-rj^{-1})} \mod p \\ &\equiv \alpha^{dr-dr} \cdot \alpha^{-rij^{-1}} \mod p \\ &\equiv \alpha^{si} \mod p \end{aligned}$$

Mivel az üzenet $x \equiv si \mod p-1$, ezért az utolsó kifejezés megegyezik az $\alpha^{si} \equiv \alpha^x \mod p$-vel, ami pontosan Aliz feltétele arra, hogy az aláírást elfogadja. A támadó az üzenet tartalmát nem tudja befolyásolni, csupán érvényes aláírást tud generálni a pszeudórandom üzenetekhez. A támadás viszont nem lehetséges, ha az üzenetet hashalték, ami a gyakorlatban gyakran előfordul. Ilyenkor az aláíró folyamatot megelőzi az üzenet hashelése és az aláíró függvény az alábbira módosul:

$$s \equiv (h(x) - \cdot r)k_E^{-1} \mod p-1$$

7. Digitális aláíró séma (DSA)

A korábban említett Elgamal séma helyett inkább a DSA-t [5] használják a gyakorlatban. Legfőbb előnye az Elgamal aláírási sémával szemben, hogy az aláírás mindössze 320 bit hosszú és néhány támadást, amivel fel lehet törni az Elgamalt, a DSA ki tud védeni.

7.1. A DSA algoritmus

Kulcs generálás

1. p prím generálása: $2^{1023} < p < 2^{1024}$
2. $p-q$-nek egy olyan q prímosztója, amire teljesül: $2^{159} < q < 2^{160}$
3. α: $\mathsf{ord}(\alpha) = q$
4. véletlenül választott d egész szám: $0 < d < q$
5. $\beta \equiv \alpha^d \mod p$ kiszámítása
6. A kulcsok: $k_{pub} = (p, q, \alpha, \beta)$, $k_{pr} = (d)$

A DSA alapötlete arra épül, hogy két ciklikus csoportot is bevonunk a számításokba. Az egyik egy nagy ciklikus csoport $\mathbb{Z}_p^*$-n és a rendje 1024 bit hosszú. A másik csoport egy 160 bites alcsoport $\mathbb{Z}_p^*$-n. Ebből adódóan maga az aláírás kevesebb bitből áll, mint az Elgamal sémánál.
Két másik bithosszúság kombináció is lehetséges p-re és q-ra nézve: $p = 2048$, $q = 448$, aláírás= 448, illetve $p = 3072$, $q = 256$, aláírás= 512.

Aláírás és hitelesítés

Úgy, mint az Elgamal aláírási séma, a DSA aláírás is két egészből tevődik össze: (r, s). Mivel mindkét paraméter csak 160 bit hosszú, ezért az aláírás maga 320 bit hosszú lesz. A publikus és titkos kulcsot használva, az alábbiak szerint írjuk alá az x üzenetet:

1. Válasszunk ki egy véletlenszerű ideiglenes kulcsot: k_E-t: $0 < k_E < q$.
2. $r \equiv (\alpha^{k_E} \mod p) \mod q$
3. $s \equiv (SHA(x) + d \cdot r)k_E^{-1} \mod q$

Az itt említett SHA hashelő függvény 160 bitesre tömöríti az x üzenetet.
Az aláírás hitelesítése:

1. segédérték 1 kiszámítása: $w \equiv s^{-1} \mod q$
2. segédérték 2 kiszámítása: $u_1 \equiv w \cdot SHA(x) \mod q$
3. segédérték 3 kiszámítása: $u_2 \equiv w \cdot r \mod q$
4. $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \mod p) \mod q$
5. Hitelesítés: $ver_{k_{pub}}(x, (r, s))$:

$$\text{ha v} \equiv r \mod q \Rightarrow \text{érvényes aláírás}$$
$$\text{ha v} \not\equiv r \mod q \Rightarrow \text{érvénytelen aláírás}$$

Amennyiben $v \not\equiv r \mod q$ nem teljesül, akkor az üzenetet vagy az aláírást módosították, vagy a hitelesítő személy nem rendelkezik a megfelelő publikus kulccsal.

Bizonyítás

$$s \equiv (SHA(x) + dr)k_E^{-1} \mod q$$

Ezt átrendezve:

$$k_E \equiv s^{-1}(SHA(x) + ds^{-1}r) \mod q$$

k_E-t másképp kifejezve:

$$k_E \equiv u_1 + du_2 \mod q$$
$$\alpha^{k_E} \mod p \equiv \alpha^{u_1+du_2} \mod p$$

Tudjuk, hogy: $\beta \equiv \alpha^d \mod p$:

$$\alpha^{k_E} \mod p \equiv \alpha^{u_1}\beta^{u_2} \mod p$$
$$(\alpha^{k_E} \mod p) \mod q \equiv (\alpha^{u_1}\beta^{u_2} \mod p) \mod q$$

Mivel $r \equiv (\alpha^{k_E} \mod p) \mod q$ és $v \equiv (\alpha^{u_1}\beta^{u_2} \mod p) \mod q$, ezért a legutolsó egyenlet: $r \equiv v \mod q$, és az aláírás érvényes.

Példa kis számokkal

Bob szeretne elküldeni egy x üzenetet Aliznak, amit DSA algoritmussal ír alá. Tegyük fel, hogy $h(x) = 26$, ami a hashelt üzenet értéke.

Aliz

Bob

$p = 59$, $q = 29$

$\alpha = 3$

$d = 7$

$\beta = \alpha^d \equiv 4 \mod 59$

$\xleftarrow{(p,qp\alpha,\beta)=(59,29,3,4)}$

Aláírás:

hash függvény:$h(x) = 26$

ideiglenes kulcs: $k_E = 10$

$r = (3^{10} \mod 59) \equiv 20 \mod 29$

$s = (26 + 7 \cdot 20) \cdot 3 \equiv 5 \mod 29$

$\xleftarrow{(x,(r,s))=(x,(20,5))}$

Hitelesítés:

$w = 5^{-1} \equiv 6 \mod 29$

$u_1 = 6 \cdot 26 \equiv 11 \mod 29$

$u_2 = 6 \cdot 20 \equiv 4 \mod 29$

$v = (3^{11} \cdot 4^4 \mod 59) \mod 29 = 20$

$v \equiv r \mod 29 \Rightarrow$ hiteles aláírás.

8. Összefoglalás

A szakdolgozatom elkészítése során megismertem a diszkrét logaritmus problémát, és hogy miért van nagy szerepe a kriptográfiában. Feltörése elég nagy prímszámok esetén sok időt vesz igénybe, még a külön erre a problémára kitalált algoritmusok segítségével is. Ezek közül az algoritmusok közül kettőt részletezek, a "kis lépés/-nagy lépés"-t és a Pohling-Hellman algoritmust.
A szakdolgozatom további részében olyan protokollokat tárgyalok, amik a DLP bonyolultságán alapszanak. Ezeket a protokollokat kulcscserére, digitális aláírásra és titkosításra is használják. A Diffie-Hellman Kulcscsere, az Elgamal séma és a DSA biztonsága is nagyban függ a protokollban használt prím nagyságától. Minél nagyobb prímszámot alkalmaznak, annál biztonságosabb az adott eljárás. Azonban megfelelően nagy prímszámot keresni nem egyszerű feladat, így szakdolgozatomban erre is kitérek. Léteznek különböző prímtesztelő eljárások, ezek elég nagy pontossággal tudják eldönteni egy számról, hogy prím-e.
Manapság a számítógépes tranzakciók a mindennapi élet részei, ezért fokozottan kell ügyelnünk adataink védelmére. Folyamatosan új algoritmusokat kell létrehozni, és ellenőrizni, hogy az eddig használt protokollok biztonságosnak tekinthetőek-e. A technológia fejlődésével ugyanis nem csak az adatainkat védő algoritmusok fejlődnek, hanem az adatainkat megszerezni kívánó támadók eszköztára is bővülhet: akár a nagyobb kapacitású gépek, vagy újabb matematikai algoritmusok megjelenésével, az eddig biztonságosnak hitt módszerek gyakorlatilag is feltörhetővé válhatnak.

9. Hivatkozások

[1] Freud Róbert és Gyarmati Edit. *Számelmélet.*

[2] Freud Róbert. *Lineáris algebra.*

[3] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.*

[4] Buttyán Levente és Vajda István. *Kriptográfia és alkalmazásai.*

[5] Christof Paar and Jan Pelzl. *Understanding Cryptography*